IBM Journal
of research and development

Vol. 28, No. 1, pp. 1–122, January 1984

A. Goyal
T. Agerwala

Performance Analysis of Future Shared Storage Systems

This paper deals with the analysis and design of two important classes of computer systems: BIP (Billion Instructions Per Second) systems consisting of a few very high performance processors and KMIP (K Million Instructions Per Second) systems with hundreds of low speed processors. Each system has large, shared semiconductor memories. Simple analytic models are developed for estimating the performance of such systems. The models are validated using simulation. They can be utilized to quickly reduce the design space and study various trade-offs. The models are applied to BIP and KMIP systems and their use is illustrated using examples.

1. Introduction

This paper reports the analyses of two generic classes of systems each consisting of several processors. The first system, called BIP, consists of a few "supercomputers" sharing a large FET memory and address space. The primary motivations are improved throughput, reliability, availability, and extendibility. Such systems will provide a total throughput in excess of one billion instructions per second. The second system, called KMIP, consists of hundreds of workstations connected to a large central memory so that individual users can share a large database. Example environments are airline reservations, insurance claims processing, computer aided design, application software development, shared document composition, etc. The primary motivation is the availability of low cost workstations in the 1 to 10 MIPS range. Computing environments in the 1990 time frame will contain elements of both systems.

BIP and KMIP are important systems that have not been reported in the literature. Since they do not exist yet, there are no empirical data about the behavior of these systems. Detailed simulation is intractable. Several design trade-offs and parameters must be studied prior to implementation. The required interconnection structure, its bandwidth and latency, and the relationship of these to processor utilization must be determined. These issues are studied by defining a queueing model of the system. The model can be simulated, but even this is expensive. By making certain approximations, the model can be solved analytically. None of the existing approximations is valid for either BIP or KMIP. A new approximation is introduced and is validated by comparing the analytic results with the results obtained by simulating the model over a large range of parameters. The approximation is then used to analyze the BIP and the KMIP systems. Using this approach, important parameters can be studied quickly. Once the trade-offs are understood and the design space reduced, more accurate information can be obtained through simulation.

In Sections 2 and 3 we discuss the systems in more detail. Existing models are presented in Section 4, and their limitations are discussed. The new approximation and validation results are given. In Sections 5 and 6 the BIP and the KMIP systems are analyzed.

2. The BIP system

The overall organization of the BIP system is shown in Fig. 1. A specific structure for achieving a billion instructions per second was studied in [1]. Eight (hypothetical) 128 MIPS processors are connected using a single gigabyte per second bus to a shared memory. Each processor has its own high speed local memory. The access time for 1000 bytes from the shared memory is 1 μs. Several portions of this system were designed and analyzed using a simulation model. High processor utilization was achieved with levels of multipro-

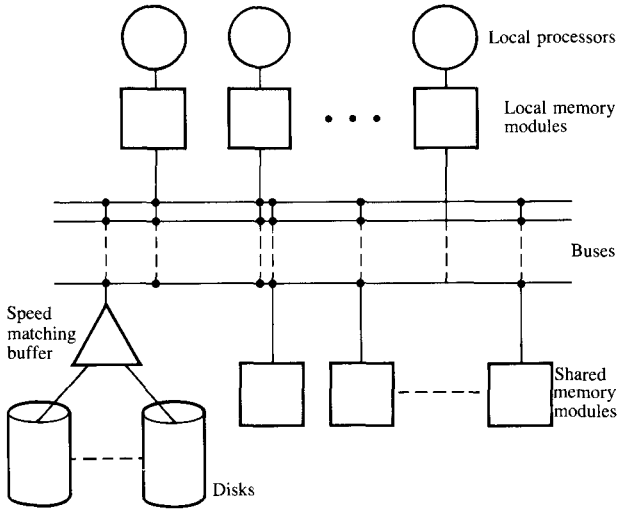

Figure 1 The BIP system.

gramming of approximately 32 per processor. This was critically dependent on a high hit ratio in the shared memory (>0.9), a high bandwidth bus of 1 gigabyte per second (GBPS), and reduced disk latency (10 ms). In the design of the system it was decided that the processor would wait synchronously for data to arrive from the shared memory. This waiting was not modeled correctly in [1] but was a negligible effect for the chosen design point. We model the effect of synchronous wait accurately in Section 4. In BIP systems, like the one described above, a large shared memory is of prime importance. We describe the main reasons below with the help of a rough calculation. Let

Number of processors	$= N$
Speed per processor	$= S \times 10^6$ instructions per second
Disk access time	$= L \times 10^{-3}$ seconds
Disk transfer rate	$= D \times 10^6$ bytes per second
Disk to memory page size	$= P \times 10^3$ bytes
Miss rate in local memory	= one in $K \times 10^3$ instructions

K is a critical parameter and must be determined experimentally. Without shared memory, for a single job, the processing time between two misses and the I/O time for a page transfer are given by

$$\text{Processing time} = \frac{K}{S} \text{ ms},$$

$$\text{I/O time} = \frac{P}{D} + L \text{ ms}.$$

Since the I/O time is tens of milliseconds and the processing time is a tenth of a millisecond (for the technologies of interest), processors must be multiprogrammed at very high levels. The degree of multiprogramming per processor, computed as the I/O time divided by the processing time, is given by

$$DOM = \left(\frac{P}{D} + L\right)\frac{S}{K},$$

and the required I/O bandwidth per job is

$$IOB = \frac{P}{\frac{P}{D} + L + \frac{K}{S}}$$

$$\approx \frac{P}{\frac{P}{D} + L} \text{ megabytes per second (MBPS)}.$$

The I/O bandwidth for the system is then

$$IOS = \frac{P}{\frac{P}{D} + L + \frac{K}{S}} \times \left(\frac{P}{D} + L\right) \times \frac{S}{K} \times N$$

$$\approx \frac{PSN}{K}.$$

The typical values for disk access time and transfer time in the 1990 time frame will be 10–30 ms and 2–6 MBPS, respectively. For $N = 10$, $S = 100$, $D = 4$, $P = 4$, $L = 20$, and $K = 10$,

$$DOM = 210,$$

$$IOS = 400 \text{ MBPS}.$$

These are fairly severe requirements. *DOM* and *IOS* are inversely proportional to K. The large shared memory (one tenth of the on-line file space) is utilized to reduce the number of I/O requests by an order of magnitude. If a fraction h of page requests are serviced by the shared memory, then

$$DOM = \left(\frac{P}{D} + L\right)\frac{S(1-h)}{K},$$

$$IOS \simeq \frac{PSN(1-h)}{K}.$$

It is expected that K will be in the range 5–20. If $h = 0.9$ can be achieved (either by proper structuring of the computation or by providing a very large shared memory), then

$$10 \leq DOM \leq 40,$$

$$20 \leq IOS \leq 80 \text{ MBPS},$$

which are more reasonable goals for the near future.

Another important issue is the block transfer size, P, on the buses. A large block size increases the latency, whereas a small block size may increase the page fault rate. If the effect of the block size on the page fault rate is known, the model

developed in Section 4 can be used to study the effect of block size on processor utilization. The overhead involved during the bus setup time can be included in the block transfer time. Other issues, such as the effect of overlapping the memory access with the bus transfer, which are not handled specifically by our model, are considered in some detail in [2].

When an access is made to the shared memory, there are two options: either the processor waits, or as in the case of disk access, it switches to another task. Task switching via an operating system call may require many instructions to be executed. In addition to this overhead, portions of the processor cache are used up by the operating system call. If the processor waits for the page request to be satisfied, a given job is executed for longer periods without operating system intervention and higher hit ratios are obtained. Some loss in performance due to waiting may be compensated by these effects. Although our analytical model can evaluate both synchronous and asynchronous waiting, some empirical data on the task switch overhead are required to make this trade-off.

3. The KMIP system

The KMIP system is based on the assumption that individual workstations of 1–10 MIPS capability will be available at very low cost in the future. Such a system is reasonable where the individual user does not require high processing speeds, but must share a database with several other users. The overall organization of the KMIP system is shown in Fig. 2. The shared memory is assumed to have enough intelligence to service page requests for the desk-top computers and to transfer data to and from the mass storage devices. Many office automation products [3] have file servers organized in similar fashion. The important difference is that in the KMIP system the data transfers occur at the paging rates rather than the occasional file transfers required in office automation systems. Relatively high bandwidth buses (2–5 MBPS) are required for the applications of interest here. Response time could be traded for communication channel bandwidths. However, the processing capacity of a 1–10 MIPS computer would not be utilized effectively. Such design trade-offs can be evaluated using our model.

The KMIP system, like the BIP system, needs a large random access shared memory, besides the main storage devices, such as disks. The important difference is that a page transfer takes place in two steps. First it is transferred on a local bus to a speed matching buffer (TSMB) and then using a local area network to the appropriate desk-top computer. Since the local network is limited in bandwidth, it is a major source of performance degradation. If the cost of the interconnection network is not prohibitive, the KMIP approach does not require the desk-top computers to have local, hard disks. In the following paragraph we give the

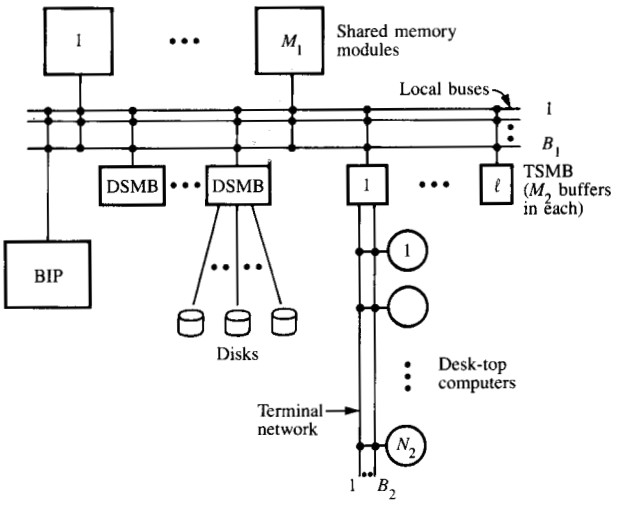

Figure 2 The KMIP system.

motivation for using a large semiconductor shared memory and show that the current technology is not suitable for a very high speed desk-top computer.

In the KMIP system all computations are performed in the desk-top computers. Since these computers are dedicated, it is assumed that they are not multiprogrammed. If all overhead except the disk access and the transfer times are ignored, the maximum average effective MIPS per processor (Me) is

$$Me = \frac{K'/S}{(P/D + L + K'/S)} \times S,$$

where $K' \times 10^3$ is the mean number of instructions between the disk requests. Without shared memory, if a 1-MIP processor missed in its local memory once every 10 000 instructions, its average processing power would be reduced to 0.25 MIPS with a 30-ms disk. With a large shared memory K' can be increased by an order of magnitude. For $K' = 100$ and $P/D + L = 30$ ms, the effective MIPS for 100-, 5-, 2-, and 1-MIP processors are 3.3, 2, 1.25, and 0.77, respectively. Thus, with current disk technology, non-multiprogrammed, inexpensive processors in the range of 1–5 MIPS are reasonable, since only 33–60 percent of the original MIPS are lost due to disk transfer time, as compared to 97 percent for a 100-MIP processor. Me is further degraded due to nonzero memory access and bus transfer times, and even further due to memory and bus interference. These effects can be evaluated using the model presented in the next section.

4. The model

Our model is constructed to study various performance and technology trade-offs and to reduce the design space using quick parametric studies. Structural questions like the num-

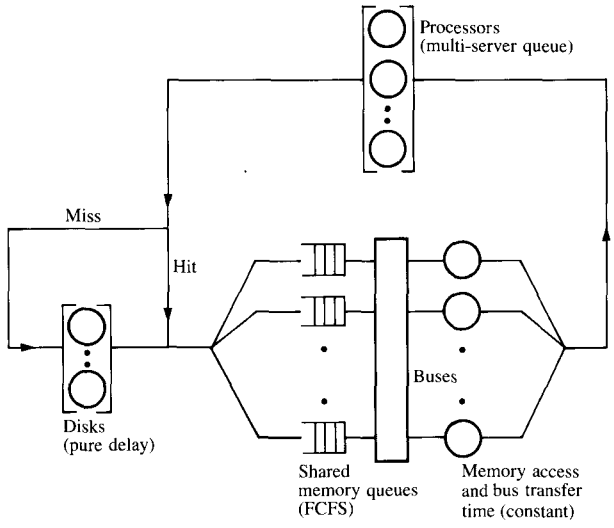

Figure 3 Queueing model for the BIP system.

ber of buses or shared memory modules to use, and technological questions like transfer rates for the bus or the shared memory for a given performance level, are some of the questions answered by the model. The desired structural inputs to the model are the number of processors (N), shared memory modules (M), and buses (B). The other system parameters are the mean processing time between page faults ($1/m$), a constant time to transfer a page from the shared memory to the processors (t), the probability that on a page fault processor i references shared memory module j (P_{ij}), the probability that a hit occurs in the shared memory (h), and some probability that a read causes a dirty page to be written back (m_w). The desired outputs of the model are processor utilization, average time spent by a page request in the shared memory system, and job throughput.

Recently, many models for similar systems have been analyzed. Bhandarkar proposed an exact solution of a system with $B \geq M$ (equivalent to a crossbar network) using a Markov chain method and showed that the state space gets very large for only a few processors [4]. He suggested an approximate solution to the problem which produced errors greater than 10 percent for certain relevant cases. Also, there was no easy way to include asynchronous writes or task switches on disk transfers. Hoogendoorn described a general memory interference (GMI) model which had an error tolerance of less than 5 percent, but he did not consider a system with an arbitrary number of buses, asynchronous writes, or task switches on disk transfers [5]. Marson and Gerla assumed the page transfer times to be exponentially distributed, which produced large errors as compared to constant page transfer times for real systems [6]. Patel applied a special case of the GMI model (equal memory reference probabilities, called UMI in this paper) to a multiprocessor system where each processor has its own private cache [7]. He did not consider an arbitrary number of buses or finite shared memory.

In this section we present a new model which satisfies all the requirements described earlier. We assume that the P_{ij}'s are all equal; however, we do mention a way to get around this problem. The first part of this section contains a *truncation* method to include buses. This is another approximation over the already existing approximations used in the GMI model. Therefore, we validate this new approximation using extensive simulations [8], some of our own and some previously reported in the literature. In the second part of this section, we show an easy way to include the effect of disk transfers by assuming the degree of multiprogramming to be infinite, and in the third part we extend our model to apply to multiple stages of page transfers to analyze the KMIP system.

• *The truncation method*

The system analyzed here is shown in Fig. 1, and its equivalent queueing model is shown in Fig. 3. Its operation and some relevant notation and assumptions are described next.

1. A processor serves a job for an average of $1/m$ time units before a *miss* in the local memory occurs. Extensive simulations done by us and in [5, 7] show that the distribution of this service time has a very small effect on processor utilization. In the following analysis only the mean service time is required, while for simulations we have assumed exponentially distributed service times with the same mean. If the time unit is one instruction execution time and geometrically distributed service times are used, then the independent probability that any instruction causes a *miss* in the local memory is m.
2. On a *miss* in the local memory, a processor references a shared memory module with an equal probability ($1/M$). The processor waits till the requested page is transferred to its local memory (synchronous wait).
3. The page request arrives at the given shared memory module in zero time. This request is actually transferred on one of the buses, and its transmission creates a small overhead. This overhead can be included in the model, but that is not the main objective here.
4. The page request waits in a queue till all the previous page requests to that shared memory module are satisfied, and a free bus is available. It takes a constant time of t units to access and transfer a page from the shared memory to the local memory.
5. The shared memory module and the bus used by this request become free and the processor starts service

again. Notice that in the above description there is no miss in the shared memory ($h = 1$); that is, none of the processors is multiprogrammed for the above mode of operation. We include $h < 1$ later.

6. Every page request causes an additional (in parallel) write request with probability m_w. Write requests follow steps (3) and (4); however, processors do not wait for their completion before resuming service.

Assume that a page request takes an average of w time units to obtain the required shared memory module and an arbitrary bus. Since the bus and the memory access algorithms do not treat a write request differently from a read request, this average wait should be common for both of them. Another point to note here is that the queueing system described here is a closed system which is always stable and has an equilibrium state. In equilibrium, the average request rate originating from any given processor is $U_p m(1 + m_w)$ pages per unit time (throughput per processor), where U_p is processor utilization. With this minimal explanation, we provide the expressions for the average number of busy memory moedules, M', and processor utilization, U_p. Applying Little's formula ($L = [\lambda][W]$) to shared memory modules, the average number of busy shared memory modules is given by

$$M' = [NU_p m(1 + m_w)][t], \tag{1}$$

and Little's formula applied to a single processor yields

$$U_p = \left[\frac{1}{\frac{1}{m} + w + t}\right]\left[\frac{1}{m}\right]$$

or

$$U_p = \frac{1}{1 + m(w + t)}. \tag{2a}$$

If processors wait for write requests to be completed,

$$U_p = \frac{1}{1 + m(1 + m_w)(w + t)}. \tag{2b}$$

Moreover, the fraction of time that a given processor has a request in the shared memory subsystem can be determined by applying Little's formula to the shared memory subsystem as follows:

$$m' = [U_p m(1 + m_w)][(w + t)]. \tag{3}$$

Equation (3) is correct whether processors wait for write requests or not. The above equations are exact and can be found in [5, 7]. Now, we have three equations (1–3) in four unknowns (M', U_p, w, and m'). Hoogendoorn approximated the above model with a discrete time model where every processor requests a memory transfer of one time unit with an independent probability of m' in every time unit. Only one request per shared memory module is satisfied, and the rest of the requests get lost. This is called Hoogendoorn's *independence* assumption here. Now, for UMI, the average number of busy memory modules can be simply calculated as follows.

In a given time unit, each shared memory module is addressed with probability m'/M from the local memory. The probability that a shared memory module is not requested is $(1 - m'/M)^N$. Conversely, the probability that a module is requested is $[1 - (1 - m'/M)^N]$. Therefore, the probability that i shared memory modules are requested is

$$\binom{M}{i}[1 - (1 - m'/M)^N]^i[(1 - m'/M)^N]^{M-i}, \tag{4a}$$

and the expected number of busy memory modules is given by

$$M' = M[1 - (1 - m'/M)^N]. \tag{4b}$$

This gives us four equations (1–4) in four unknowns (M', U_p, w, and m'), which can be reduced to a single nonlinear equation in U_p. Therefore, the performance of the system can be evaluated by solving this nonlinear equation. Both Hoogendoorn and Patel observed that the above approximation for evaluating M' and U_p matches very well with the simulation results. Equation (4) becomes quite complex for nonuniform reference probabilities. M' can still be evaluated by using an iterative method as suggested in [5]. Although the GMI model can be extended to model bus contention, we show only the UMI extension.

The probability that i ($1 \le i \le M$) shared memory modules are requested is given in Eq. (4a), which is a binomial probability distribution. If $B \ge M$, then Eq. (4a) also gives the probability that i ($1 \le i \le M$) shared memory modules are busy. However, if $B < M$, the maximum possible number of busy memory modules is B. Therefore, the average number of busy memory modules for an arbitrary number of buses is approximated as

$$M' = \sum_{i=0}^{M}\binom{M}{i}[1 - (1 - m'/M)^N]^i \times [(1 - m'/M)^N]^{M-i} \min(i, B), \tag{5}$$

and the utilization of an individual bus is given by

$$U_B = M'/B. \tag{6}$$

This simple method for redistributing the probabilities is called *truncation* in this paper. *Truncation* is a very simple extension of the *independence* approximation already made by Hoogendoorn. Since the *independence* assumption has been validated for a large number of test cases in [5, 7], we expect it to work well with the *truncation* extension also.

Table 1 Average number of busy memory modules with $N = M = 16$, no writes, and infinite shared memory; (a) our simulation, (b) analysis, and (c) Lang's simulation.

Mean processing time		*1*	*2*	*4*	*8*	*16*	*32*
Buses							
1	a	1	1	1	1	0.8584	0.4794
	b	1	1	1	0.9997	0.8719	0.4811
	c	1					
2	a	2	2	2	1.6616	0.9316	0.4852
	b	2	2	1.9963	1.6655	0.9331	0.4847
	c	2					
3	a	3	3	2.7924	1.7547	0.9391	
	b	3	2.9953	2.7810	1.7479	0.9396	
	c	3					
4	a	4	3.9581	3.0523	1.7667		
	b	3.9943	3.9200	3.0368	1.7660		
	c	4					
5	a	4.9889	4.6138	3.1146			
	b	4.9553	4.5467	3.1056			
	c	4.98					
6	a	5.8796	4.8807	3.1288			
	b	5.7856	4.8384	3.1337			
	c	5.85					
7	a	6.4716	4.9567				
	b	6.3743	4.9576				
	c	6.43					
8	a	6.7388	4.9848				
	b	6.7174	5.0007				
	c	6.70					
9	a	6.8144	5.0083				
	b	6.8795	5.0206				
	c	6.82					
10	a	6.8404					
	b	6.9499					
	c	6.83					
11	a	6.8536					
	b	6.9758					
	c	6.83					
12	a	6.8469					
	b	6.9825					
	c	6.83					

Table 2 Processing power for $N = 6$, $M = 4$, $B = 2$.

ρ	$P_{\text{sim.}}$	$P_{\text{ana.}}$
0.001	5.99	5.994
0.01	5.94	5.940
0.1	5.42	5.418
0.333	4.14	4.164
0.5	3.37	3.384
0.75	2.49	2.508
1.0	1.95	1.944
3.0	0.66	0.666
5.0	0.40	0.402

A large number of simulations were run to validate the above model. These simulations were run long enough so that 90 percent confidence intervals for M' and U_p had relative half-widths of less than 2 percent.

Example 1

a. $N = M = 16$, B is increased from 1 to 16.
b. $m_w = 0$ (no write requests).
c. The sum of the memory access time and the bus transfer time is assumed to be constant at 1 time unit, i.e., $t = 1$.
d. The mean processor service time is varied from 1 to 32 in multiples of 2.

Table 1 shows the average number of busy memory modules obtained from the simulation (a) and the analysis (b). Processor utilization can be obtained from Eq. (1). All analytical results are within 2 percent of the means obtained from simulation. The average number of busy memory modules increases linearly in the beginning (region 1) as the number of buses increases. It tapers off very quickly and does not change any further (region 2). We truncate each column in the table when increasing the number of buses does not change the analytical value of M' in the third place after the

Table 3 Average number of busy memory modules with $m_w = 0.3$, $N = M = 16$, and infinite shared memory; (a) simulation, (b) analysis.

Mean processing time		*1*	*2*	*4*	*8*	*16*	*32*
Buses							
1	a	1	1	1	1	0.9683	0.6138
	b	1	1	1	1	0.9939	0.6197
2	a	2	2	2	1.9036	1.1904	0.6312
	b	2	2	2	1.9554	1.2059	0.6310
3	a	3	3	2.9924	2.2116	1.2155	0.6332
	b	3	3	2.9882	2.2361	1.2165	0.6330
4	a	4	3.9981	3.6629	2.2733	1.2185	
	b	4	3.9968	3.6964	2.2771	1.2205	
5	a	4.9989	4.9451	3.9007	2.2848		
	b	4.9978	4.9499	3.9352	2.2830		
6	a	5.9824	5.6633	4.0034	2.2898		
	b	5.9819	5.6866	4.0082	2.2928		
7	a	6.8936	6.0589	4.0265			
	b	6.8980	6.0899	4.0331			
8	a	7.6089	6.1989	4.0314			
	b	7.6525	6.2725	4.0413			
9	a	8.0455	6.2844				
	b	8.1590	6.3475				
10	a	8.2419	6.3154				
	b	8.4420	6.3772				
11	a	8.2984	6.3204				
	b	8.5757	6.3840				
12	a	8.3429					
	b	8.6295					
13	a	8.3421					
	b	8.6490					

decimal. In the first region, buses are the bottleneck, and in the second region, memory interference is the bottleneck. Notice that when the value of $1/m$ is large, the buses are never a bottleneck. If there were no bottlenecks, the maximum processor utilization would be that obtained from Eq. (2) by setting w to zero. The analytic and simulated probability distribution for the number of busy memory modules is given in [2], and it is concluded that although the simulated probability distribution is not exactly the truncated binomial, the probability mass does tend to stack up at B and its mean is very accurate (Table 1), as predicted in the *truncation* approximation.

In the first column of Table 1, we have included some simulation statistics (c) for M' obtained by Lang et al. [9]. Besides simulation variations, a small discrepancy between their simulation statistics and ours is due to the fact that they model the processor service time ($1/m$) using a geometric distribution with mean 1, whereas we used an exponential distribution with the same mean. This observation also shows that the processor service time distribution has little effect on the performance of the system shown in Fig. 1.

In [6], Marson and Gerla obtained some simulation results for multiple bus architectures. For constant memory access and bus transfer times, they obtained *processing power* which is equivalent to NU_p in our case. The value of ρ used there is equivalent to the product mt. We compare our analytical results for a system with $N = 6$, $M = 4$, and $B = 2$ to their simulation results [6, Table IV], in our Table 2. Time t is assumed to be constant at 1, and m is varied from 0.001 to 5. The analytical results are within 1 percent of their simulation.

Example 2

Same as Example 1, except that $m_w = 0.3$.

This increases the memory interference and decreases processor utilization [Eq. (2b)]. When memory interference is extremely high, the *independence* assumption is not very good. This is evident from the fact that the largest relative error is in the last entry of the first column of Table 3, where there is no bus interference and consequently the truncation extension does not affect the analytical results and the error is due to the *independence* assumption only. However, the relative errors are still less than 4 percent. (Similar observation is made after we include the disk transfers.) Since most real system designs tend to keep processor utilization very high (or average number of busy memory modules low), our analysis is quite pragmatic.

Patel [7] presents several local memory organizations, such as buffered write-back, write-through, etc. Different

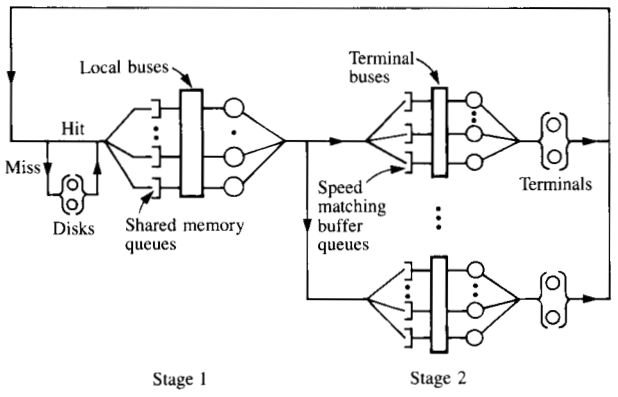

Figure 4 Queueing model for the KMIP system.

organizations require appropriate changes in the expressions for U_p, M', and m', whereas the equation for the equivalent model, Eq. (5), remains the same. In the following subsections we present some useful models for finite/infinite shared memory and synchronous/asynchronous waits on read/write requests.

• *Infinite shared memory/asynchronous wait*

In this case, processors do not even wait for the read requests to be completed. As soon as a *miss* in the local memory occurs, they start working on another job. To keep the analysis simple, we assume an infinite supply of jobs at the processor. In other words, processors are not idle because of a lack of jobs. A job switch causes a certain amount of overhead (e.g., for I/O routines, dispatching, etc.) which takes, say, t_o units of time. Therefore, the processor utilization becomes

$$U_p = \frac{1}{1 + mt_o}, \tag{7}$$

and the expression for M' remains the same as Eq. (1). Equations (1) and (7) can be solved exactly to obtain U_p and M'. Hence, no validation is required.

• *Finite shared memory/synchronous wait at shared memory/synchronous wait at disk*

This model is useful in analyzing the KMIP system described in Section 3. A busy processor requests a shared memory transfer with probability mh and a disk transfer with probability $m(1 - h)$ in every time unit. The time to service a shared memory transfer is $(w + t)$, and a disk transfer is

$$t_d = \left(w + L + \frac{k'}{D'} + k't\right) + (w + t), \tag{8}$$

where

L = latency + seek,

k' = number of pages transferred from the disk subsystem,

D' = disk speed in pages per unit time.

The first term in t_d is for the transfer of a block of k' pages from the disk to the shared memory, and the second term is for the transfer of a single page from the shared memory to the local memory of the requesting processor. Therefore, the processor utilization becomes

$$U_p = \frac{1}{1 + mh(w + t) + m(1 - h)t_d}. \tag{9}$$

Processor wait on writes can be included by multiplying m with $(1 + m_w)$ in the above equation. Similarly, the average number of busy memory modules can be calculated as

$$M' = NU_p m[1 + m_w + (1 - h)k']t. \tag{10}$$

The equivalent unit request rate on the network is evaluated by including the traffic created by disks. Therefore, the equation for m' becomes

$$m' = U_p m[1 + m_w + (1 - h)k'](w + t). \tag{11}$$

The above three equations, together with Eq. (5), can be solved to evaluate system performance. The validation of this disk model is done later in this section.

• *Finite shared memory/synchronous wait at shared memory/asynchronous wait at disk*

Now a processor does not wait for disk transfers. This model is used to analyze the BIP system. Again to keep the analysis simple, we assume an infinite supply of jobs at the processor. Processor utilization then becomes

$$U_p = \frac{1}{1 + m(w + t) + m(1 - h)t_o}, \tag{12}$$

and the terms for M' and m' used in the last subsection are still valid. The analytical results are still within 5 percent of the simulation as evident from the tables given in [2]. If a processor does not wait for read transfers also, Eq. (7) for U_p and Eq. (10) for M' describe the exact system behavior.

• *Multiple stages of access*

It is possible to extend the above model to multiple stages of access and transfer. Some multiprocessing systems have two or more stages in which the accesses and transfers take place. For example, in the KMIP system shown in Fig. 2, a page is first read from the shared memory to a speed matching buffer using a local bus, and then this page is transferred to the requesting processor using a local area network. The equivalent queueing model for the KMIP system is shown in Fig. 4. We denote the first and the second stages by subscripts 1 and 2, respectively. Any stage can be modeled by any one of the four models presented up to this point. Specifically, the first stage will use the model with synchro-

nous waits at the disk and the second stage will use the model with infinite shared memory and synchronous wait. Assume that all the parallel branches in the second stage are identical. The average number of busy TSMBs (speed matching buffers) per local area network, M'_2, can be written as

$$M'_2 = N_2 U_p m(1 + m_w)t_2, \tag{13}$$

where N_2 is the number of processors on each local area network, and t_2 is the transfer time for a single page on this network. The corresponding m'_2 is given by Eq. (3), where w and t are replaced by w_2 and t_2, respectively. The average number of busy shared memory modules, M'_1, is given by

$$M'_1 = N_2 \ell U_p m[h + m_w + (1 - h)k']t_1, \tag{14}$$

where ℓ is the total number of parallel branches in stage 2 (total number of local area networks). We use h instead of 1 [see Eq. (10)] inside the parentheses because we assume that a page transferred from the disk is received by both the shared memory and the speed matching buffer simultaneously. The corresponding equation for m'_1 is therefore

$$m'_1 = U_p m[h + m_w + (1 - h)k'](t_1 + w_1), \tag{15}$$

and the equation for U_p is given by

$$U_p = \frac{1}{1 + m\left[w_2 + t_2 + h(w_1 + t_1) + (1 - h)\left(w_1 + L + \frac{k'}{D'} + k' t_1\right)\right]}. \tag{16}$$

The equivalent model [Eq. (5)] can be applied to stages 1 and 2 in succession to yield two nonlinear equations in w_1, w_2, and U_p. These equations, together with Eq. (16), can be solved numerically to evaluate the performance of the system. We programmed a very simple bisection method in 64-bit arithmetic to solve the above equations. The solution did not converge for cases where utilizations for both networks (U_{B1}, U_{B2}) were extremely close to one (>0.99). In all other cases, the solution converged very quickly. The number of iterations was substantially smaller when we applied the bisection method to the less loaded (low bus utilization) stage first. The details of the experiments performed on the KMIP system are given in [2]. All utilization statistics were estimated within four percent of the means obtained via simulation.

There are two main contributions of the above models. First, limited-bus architectures can be analyzed, and second, multiple stages of access and transfer can be accommodated. None of these have been previously reported in the literature. Experimentation shows that most of the errors introduced in the above model are due to the memory interference model chosen, and not because of the *truncation* extension. Had we applied the *truncation* extension to Rau's model [10] (zero processing time), more accurate results could have been obtained. However, within the regions of interest ($U_p > 0.5$ for the BIP and U_{B1}, $U_{B2} < 0.9$ for the KMIP), the above models can be used to analyze the BIP and the KMIP systems with sufficient accuracy.

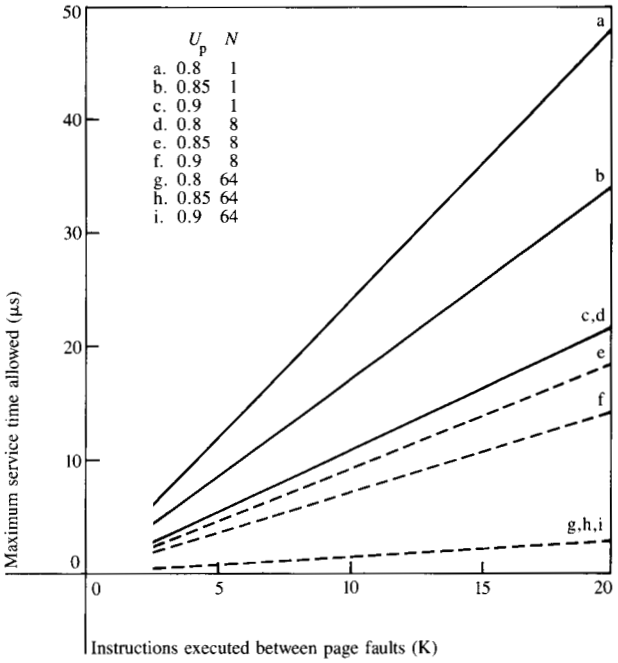

Figure 5 Analysis of a BIP system with 100 MIPS/processor, 4K bytes/page, $m_w = 0.3$, and $M = B = 1$.

5. Analysis of the BIP system

The interconnection structures considered here are crossbar and multiple buses. Our primary concern here is the bandwidth that should be provided for a single block transfer from the shared memory to a processor. The effect of writing back dirty pages is included in the model, as described in Section 4. All graphs drawn in this section assume that 0.3 page is written back for every page read. The overhead is included in the transfer time t and consists of several delays: determining where the block is located in the shared memory, establishing the connection, and initiating the access. The overhead is dependent on the technology and the complexity of the interconnection network and must be determined for each system. For the BIP system [1] this overhead was approximately 1 μs. This section discusses system design issues in the case where the processor waits for a shared memory transfer.

- *Infinite shared memory*

Given the average number of instructions executed between page faults (K) and the speed of the processor (S), the mean

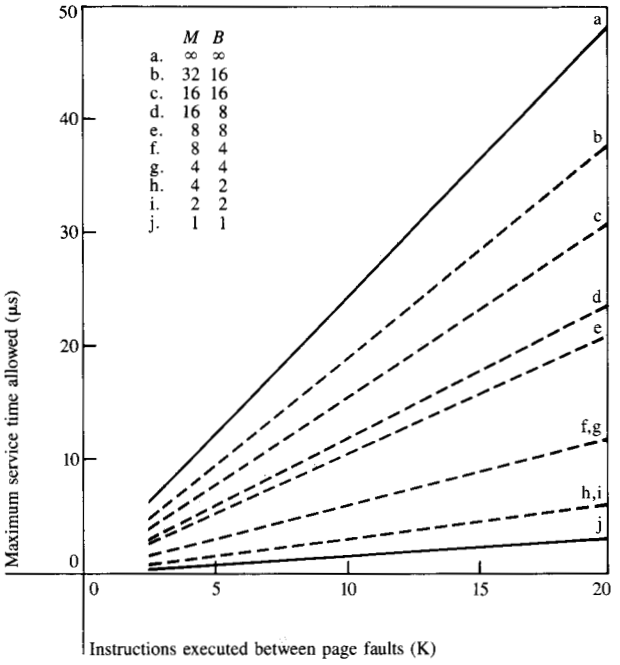

Figure 6 Analysis of a 64-processor system with 100 MIPS/processor, 4K bytes/page, $m_w = 0.3$, and $U_p = 0.8$.

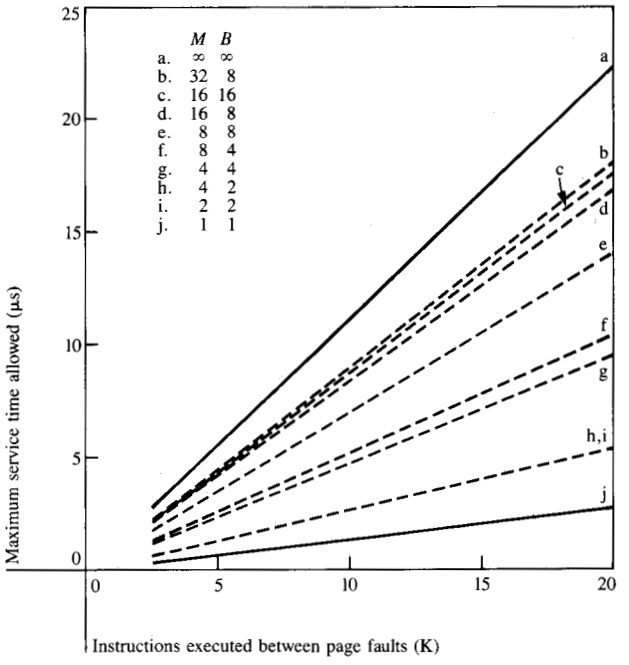

Figure 7 Analysis of a 64-processor system with 100 MIPS/processor, 4K bytes/page, $m_w = 0.3$, and $U_p = 0.9$.

arrival rate of requests, m, from a processor to the shared memory subsystem is known. For different system structures with varying numbers of buses (B) and memory modules (M), the model in Section 4 provides the processor utilization (U_p). The response time has two components, the average wait time (w) and the fixed service time (t). Since we are interested in the latency requirements, results are plotted as shown in Fig. 5, i.e., maximum service time (t) allowed vs K, for fixed processor utilization.

Curves a, b, and c are for $U_p = 0.8$, 0.85, and 0.9, respectively, for $S = 100$, and for $N = 1$. Since there is no interference, these curves place an upper bound on the service time (t) and, therefore, a lower bound on the technology. Curves d, e, and f are for $M = B = 1$ and $N = 8$. At $U_p = 0.9$ and $K = 10$, the maximum service time allowed is 6 μs. Assuming 1 μs as overhead, the access and transfer must take place within 5 μs. Therefore, for a 4K-byte page size, the bus bandwidth must be at least 800 MBPS, and even if the memory access and the bus transfers are overlapped per 1000 bytes, the memory access time must be less than 1 μs. This example illustrates the following important point: Synchronous paging with low performance degradation places severe requirements on the bus technology. Figure 5 also gives the average wait time (w) for a fixed U_p and K as explained below.

At $K = 10$ and $U_p = 0.9$, from curve d, t is 10 μs. The wait time is therefore 4 μs. Curves g, h, and i are for $N = 64$ and $M = B = 1$. Clearly, the response time in this system is dominated by the wait time. The required service time cannot be supported with the available technology. In this case, w must be reduced by increasing M and B. The precise effect is shown in Figs. 6 and 7 for $U_p = 0.8$ and 0.9, respectively. From Fig. 7, at $K = 10$, at least 8 memories and 8 buses are needed to obtain $t = 6$ μs, each of which will have the same requirements as mentioned in the previous paragraph, provided the overhead remains at 1 μs.

Figure 7 also illustrates that the system is either memory limited or bus limited at different values of M and B. At $(M, B) = (2, 2)$, the system is completely bus limited, and increasing M to 4 has no effect on t. Average wait time w is substantially reduced in moving from (4, 2) to (4, 4). At (4, 4) the system is not bus limited; increasing M from 4 to 8 reduces w. At $B = 8$ the system is no longer bus limited; increasing M from 8 to 16 to 32 decreases w continuously. For $N = 64$, $S = 100$, the appropriate system structure is $M = 16$ and $B = 8$. The analysis of a 16-processor system is described in [2], and it is concluded that, in general, $N/4$ buses and $N/2$ memories represent a good cost-effective design point. Based on availability considerations, a constant number of buses and memory modules may be added.

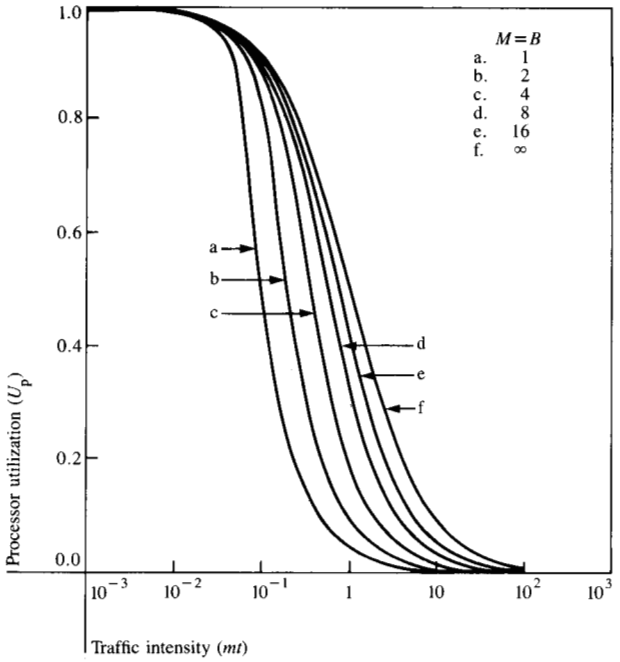

Figure 8 Analysis of a 16-processor system with 100 MIPS/processor, 4K bytes/page, $m_w = 0.3$, and $t = 5$ μs.

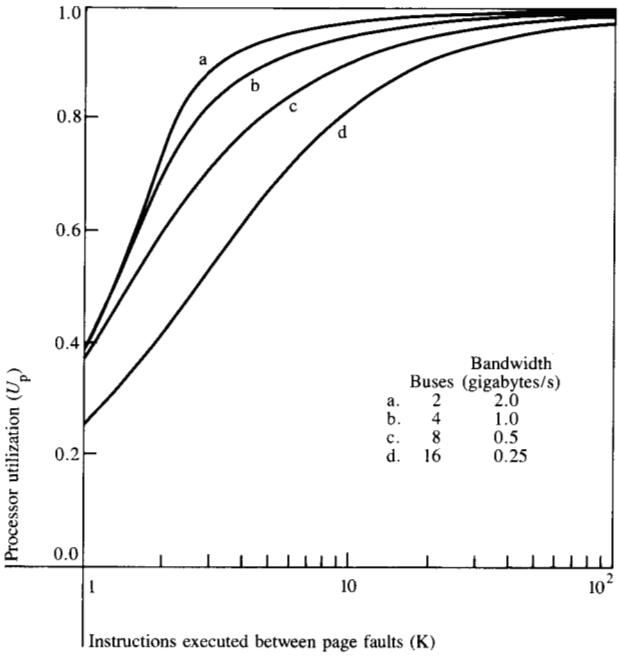

Figure 9 Analysis of a 16-processor system with 100 MIPS/processor, 4K bytes/page, $m_w = 0.3$, and network bandwidth = 4 gigabytes/second.

The latency requirements on the bus and memory can be reduced to some extent by using certain design techniques. The interested reader is referred to [2] for details. In general, t must be chosen so that degradation is maintained within certain bounds, since K will be specified as a range. In Fig. 8, U_p is plotted against traffic intensity, the product mt. Time t is chosen to be 5 μs, so that the expected traffic intensity for a range of K (5 to 20) is below 10^{-1}. At higher levels, performance is too sensitive to K. In Fig. 9, we have plotted processor utilization vs K, keeping the total available bus bandwidth at 4 GBPS. All curves are for a system in which $N = M = 16$. For nominal values of K (5 to 20), four buses of one gigabyte each are recommended.

- *Finite shared memory*

In the previous analysis, page transfers between local and shared memories were included, but I/O transfers were completely ignored. In some environments, the required I/O bandwidth can be large. Since I/O transfers utilize the bus and the shared memory resources, their effect cannot be neglected.

In Section 4, the execution model was extended to include the effect of I/O transfers. I/O bandwidth was considered, but it was assumed that the degree of multiprogramming was large enough so that there was no degradation in performance due to I/O latency. (I/O latency affects utilization and bus bandwidths if processors wait synchronously for the disk transfers. Such systems are considered in the next section.) This model gives a lower bound on t. The infinite shared memory analysis provided the upper bound. For a given level of performance the overall design space can now be substantially constrained.

Figure 10 gives the effect of disk transfer on t for $N = 16$, $U_p = 0.9$, and $K = 10$. The minimum service time required for 4K- and 64K-byte disk transfers is plotted against the mean number of instructions between page faults in shared memory. If the x axis is extended to infinity, shared memory page faults go to zero, which is equivalent to infinite shared memory. For a given miss ratio a page size of 64K bytes always require higher bus bandwidth than a page size of 4K bytes. If the miss ratio is reduced by 16 times in going from 64K bytes to 4K bytes, their corresponding latency requirements, t, become comparable.

The BIP systems studied in this section are fairly complex, consisting of multiple processors, memories, buses, and disks. These systems have been modeled using a very simple approach which includes the effects of synchronous wait on shared memory transfers and asynchronous wait on I/O transfers. The model predicts (as expected) that bus latency

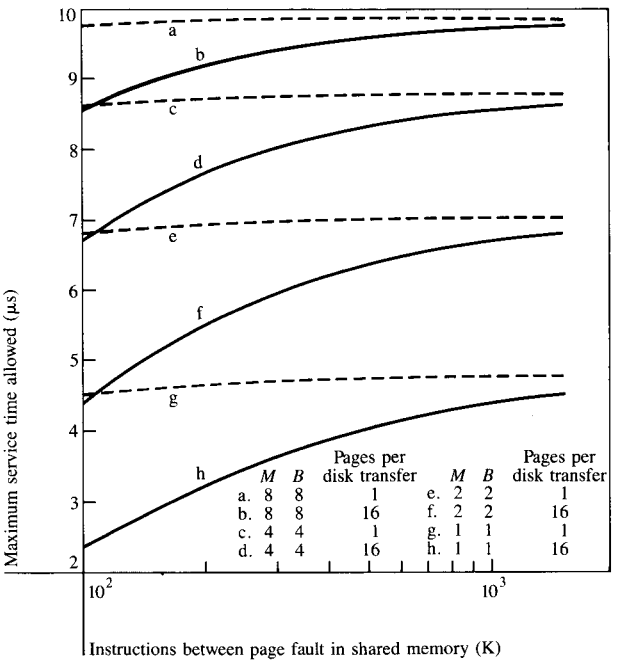

Figure 10 Analysis of 16-processor system with 100 MIPS/processor, 4K bytes/page, $m_w = 0.3$, and $U_p = 0.9$.

is critical and allows one to study the precise trade-offs between latency per bus and the number of buses and memories. The model can be used to study performance degradation due to failures which result in the loss of buses and memories. In designing BIP systems, hit ratios in local and shared memory are critical parameters which must be estimated for each environment. Once these parameters are known, for given performance levels, the design space can be substantially reduced using the models developed. Subsequently, more detailed analysis and simulation can be attempted for specific implementations.

6. Analysis of the KMIP system

In this section the bandwidth requirements of the KMIP interconnection network are determined. The analysis of the KMIP system involves two steps. First, the concept of an *active user* is introduced, and all subsequent analysis is based on the number of active users. In KMIP, it is assumed that on a miss in shared memory, a 4K-byte page is first transferred from the disk to the shared memory and the speed matching buffer, and then from the speed matching buffer to the requesting computer. In step 2, we use the multiple stages of access model from Section 4 to evaluate the latency requirements on both the local and the long buses for a given level of processor utilization. The maximum possible utilization may be found by setting average wait times in Eq. (16) to zero. It is shown that the long buses are the bottlenecks for achieving effective MIPS close to the potential, when the number of active users is large. The effect of increasing the number of access ports per desk-top computer is discussed briefly. An example is given to illustrate design trade-offs.

- *Active users*

A computer is *active* when an instruction stream is executed locally in response to a single user command, and it generates page faults at the rate of one every 10K–100K instructions. As long as a user is *thinking,* he is not active. Requests for file transfers for editing, storing, back-up, and journaling do not create an active user. One study [11] shows that the number of active users can be anywhere from one in 10 to one in 50 installed terminals (in peak hours) even in scientific environments. That is, for every active user supported on the system, 10–50 desk-top computers should be allowed to use the system to optimize cost vs performance. Our subsequent analysis only deals with active users since they place the most stringent requirements on the network. For a given environment, a specific ratio must be obtained empirically to compute the total number of desk-top computers to be installed. In the following section, we give an example to show how the model of Section 4 can be used to evaluate KMIP systems.

- *Numerical example*

Given 100 personal computers of 2 MIPS capability each, design a KMIP system.

The first step is to find what bus bandwidths are feasible over the required distances. Assume that a maximum of 10 MBPS and 100 MBPS are reasonable for the long and the short buses, respectively. Some empirical data, such as hit ratio in the shared memory ($h = 0.9$) and the number of instructions (10K) processed before a page-fault in the local memory, are also given. The second assumption implies that $1/m = 5$ ms. Assume that it takes an average of 30 ms for disk latency and seek and 1 ms to transfer a 4K-byte page. Given these data, one can evaluate Me ($= 1.25$) from the equation given earlier in Section 3, which implies that processor utilization cannot exceed 0.625. Assuming that we are satisfied with a processor utilization of 0.5, what values of N_2, ℓ, and B_1 should be used?

Let us find the total bandwidth requirements for each of the stages. Since we know $N_2\ell = 100$, the number of page requests made per second are known:

$$N_2\ell U_p m = 10\,000.$$

Assuming that for every 4K-byte page request there is 1K bytes of overhead, the total bandwidth requirement for each stage is 50 MBPS. Let us provide twice that amount in each stage to keep the wait times small, which gives $\ell = 10$, $N_2 = 10$, and $B_1 = 1$. Now, the last model of Section 4 can be

used to evaluate the performance of the above configuration:

$U_p = 0.563, (0.562)$

$w_1 + t_1 = 73.4, (72.85)\ \mu s,$

$w_2 + t_2 = 705.5, (744.4)\ \mu s,$

$U_{B2} = 0.563, (0.562).$

The above system has been over-designed. One could reduce the bandwidth of each long bus, or could reduce the number of long buses without going below a U_p of 0.5. A few iterations would be required to satisfy all the design requirements. The approach outlined in the previous paragraph gives a satisfactory starting point. We simulated the above configuration to increase the reader's confidence in our model. The simulation results are given above (in parentheses).

In this section we analyzed systems consisting of hundreds of slow speed processors (1–5 MIPS) connected via time-shared buses to a large shared memory. The motivation for considering such systems was given. The focus was on the interconnection network requirements. A 2-MIP desk-top computer is degraded to approximately 1.25 MIPS due to waiting synchronously for disk transfers (for typical page sizes and hit ratios). After this effect, the most important factor is the latency of the long buses. Based on the models developed in Section 4, latency requirements were obtained for these buses given certain levels of further performance degradation. For example, a 2-MIP processor will be degraded to 1.13 MIPS if 10-MBPS-long buses are used with 10 active users each. (Note: 10 active users does not imply 10 desk-top computers per long bus.) As in Section 5, models are used to obtain only the first-order estimates and to reduce the overall design space.

7. Conclusions

Simple models were developed for studying several important computer systems consisting of processors, buses, and shared memory modules. The models used the *independence* approximation suggested by Hoogendoorn. A further approximation, called *truncation*, was introduced to analyze multiple bus systems. The analytical results were within 5 percent of the simulation results over the design space of interest. The models were applied to BIP and KMIP systems. It was shown that for specified levels of performance the models can be used to quickly reduce the design space, after which more accurate (and expensive) techniques can be applied to obtain better performance estimates.

The results indicate that the bandwidth of a single bus is the most critical parameter. If the required bandwidth cannot be supplied on a single bus due to technological constraints, it may be traded to some extent with an increased number of buses and shared memory modules, as shown in this paper. Since the number of buses is a parameter in our models, the effect of bus failures on system performance may be studied by varying this parameter. This is in contrast to the previous analytical models which assumed a fully functional interconnection network (crossbar) during their development.

Acknowledgments

We would like to thank John Cocke for motivating us to study BIP and KMIP systems and Don Towsley, Phil Heidelberger, and Steve Lavenberg, for numerous discussions and suggestions.

References

1. Evan M. Tick, "Design and Analysis of a Memory Hierarchy for a Very High Performance Multiprocessor Configuration," Master's Thesis, Massachusetts Institute of Technology, Cambridge, MA, January 1982.
2. A. Goyal and T. Agerwala, "Design Considerations in Some Shared Memory Systems," *Research Report RC-10028*, IBM Thomas J. Watson Research Center, Yorktown Heights, NY, June 1983.
3. A. Goyal and M. Malek, "Local Area Networks: A Status Report," *Technical Report*, Department of Electrical Engineering, University of Texas, Austin, TX, December 1981.
4. D. P. Bhandarkar, "Analysis of Memory Interference in Multiprocessors," *IEEE Trans. Computers* **C-24,** 897–908 (September 1975).
5. C. M. Hoogendoorn, "A General Model for Memory Interference in Multiprocessors," *IEEE Trans. Computers* **C-26,** 990–1005 (October 1977).
6. M. A. Marson and M. Gerla, "Markov Models for Multiple Bus Multiprocessor Systems," *IEEE Trans. Computers* **C-31,** 239–248 (March 1982).
7. J. H. Patel, "Analysis of Multiprocessors with Private Cache Memories," *IEEE Trans. Computers* **C-31,** 296–304 (April 1982).
8. *Computer Performance Modeling Handbook*, S. S. Lavenberg, Ed., Academic Press, Inc., New York, 1983.
9. T. Lang, M. Valero, and I. Alegre, "Bandwidth of Crossbar and Multiple-Bus Connections for Multiprocessors," *IEEE Trans. Computers* **C-31,** 1227–1234 (December 1982).
10. B. R. Rau, "Interleaved Memory Bandwidth in a Model of a Multiprocessor Computer System," *IEEE Trans. Computers* **C-28,** 678–681 (September 1979).
11. E. Fuchs and P. E. Jackson, "Estimates of Distributions of Random Variables for Certain Computer Communications Systems," *Proceedings*, Symposium on Problems in the Optimization of Data Communications Systems, Pine Mountain, GA, October 1969, pp. 202–225.

Received June 21, 1983; revised September 27, 1983

Tilak Agerwala *IBM Research Division, P.O. Box 218, Yorktown Heights, New York 10598.* Dr. Agerwala is senior manager of the computation-intensive systems group at the Thomas J. Watson Research Center. He received a B.Tech. from the Indian Institute of Technology, Kanpur, in 1971 and a Ph.D. from The Johns Hopkins University, Baltimore, Maryland, in 1975, both in electrical engineering. From 1975 to 1979, he was an assistant professor in the Departments of Electrical Engineering and Com-

puter Sciences at the University of Texas at Austin. Dr. Agerwala joined the IBM Research Division in 1979. From 1980 to 1983 he was manager of the architecture and systems design group. In this capacity he led a research effort on high-performance computers for scientific/engineering applications. He received an invention award and an outstanding technical achievement award from IBM for work in this area. His research interests are in computer architecture, super-computers, and parallel processing. Dr. Agerwala is a senior member of the Institute of Electrical and Electronics Engineers and a Distinguished Visitor of the IEEE Computer Society.

Ambuj Goyal *IBM Research Division, P.O. Box 218, Yorktown Heights, New York 10598.* Dr. Goyal is a member of the Computer Science Department at the Thomas J. Watson Research Center. His research interests include analytical and design problems associated with multiprocessor systems, storage hierarchies, digital communications, and reliable systems. He received his B.Tech. degree from the Indian Institute of Technology, Kanpur, in 1978, and the M.S. and Ph.D. degrees from the University of Texas at Austin in 1979 and 1982. Dr. Goyal is a member of the Institute of Electrical and Electronics Engineers and its Computer Society.